Compact crypto implementations for embedded security
or
crypto + embedded systems = security?

Ingrid Verbauwhede
KULeuven – COSIC

E-mail: ingrid.verbauwhede@esat.kuleuven.be

Slide Acknowledgements:
Current & former Ph.D. students

Outline

IEEE Advancing Technology for Humanity

Benelux Embedded Systems chapter

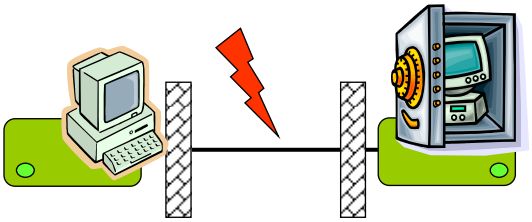
+

BCRYPT

IUAP – Belgian Fundamental Research on Cryptology and Information Security

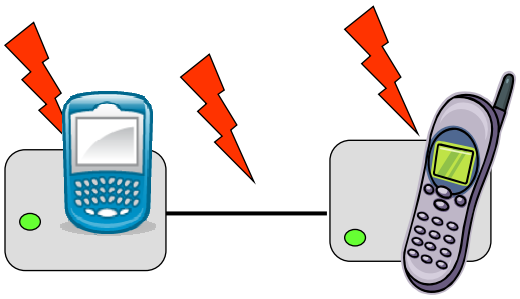
- Merge: Embedded systems & Security
- Definition: secure embedded systems
- Illustrate with examples
- Challenges
- Conclusions

Embedded security: definition (1)



Old Model (simplified view):
-**Attack on channel between communicating parties**
-**Encryption and cryptographic operations in black boxes**
-**Protection by strong mathematic algorithms and protocols**

Embedded security: definition (2)



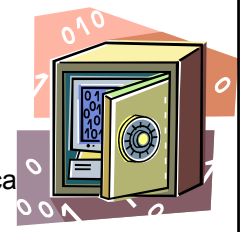
New Model (also simplified view):
-**Attack channel *and* endpoints**
-**Encryption and cryptographic operations in gray boxes**
-**Protection by strong mathematic algorithms and protocols**
-**Protection by secure implementation**

Need secure implementations not only algorithms

Embedded Security: definition (3)

NEED BOTH

- Efficient, lightweight implementations
  - Within power, area, timing budgets
  - Public key: 2048 bits RSA, 200 bit ECC on 8 bit μC and 100 μW
  - Public key on a passive RFID tag



- Trustworthy implementation
  - Resistant to attacks
  - Active attacks: probing, power glitches, JTAG sca
  - Passive attacks: side channel attacks

Illustrate with examples

- Example 1: Secret Key: KATAN, KTANTAN
- Example 2: NIST SHA3 – how not to do it
- Example 3: Public key for RFID

Secret key: KATAN, KTANTAN

Christophe De Cannière, Orr Dunkelman and Miroslav Knežević

CHES 2009

[slide courtesy: M. Knežević]

KATAN/KTANTAN Design Goal

- *Minimum* logic (i.e. gates) to implement a secret key algorithm

Alternatives:

- Stream ciphers
  - To ensure security, the internal state must be twice the size of the key.
  - No good methodology on how to design these.
- Use a standardized block cipher: AES
  - The smallest implementation consumes 3.1 Kgates.
  - Designed for HW and SW implementations
- Other block ciphers?
  - HIGHT, mCrypton, DESL, PRESENT,…
  - Can we do better/different?

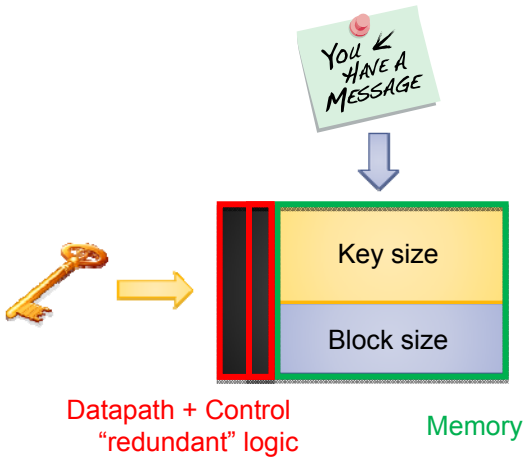
Design Goals

- Secure block cipher
  - Address Differential/Linear cryptanalysis, Related-Key/Slide attacks, Related-Key differentials, Algebraic attacks.
- Efficient block cipher
  - Small foot-print, Low power consumption, Reasonable performance (+ possible speed-ups).
- Application driven
  - Does an RFID tag always need to support a key agility?
  - Some low-end devices have one key throughout their life cycle.
  - Some of them encrypt very little data.
  - Tune algorithm to the application!

KATAN/KTANTAN Block Ciphers

- Block ciphers based on Trivium (its 2 register version – Bivium).
- Block size: 32/48/64 bits.
- Key size: 80 bits.
- Share the same number of rounds – 254.
- KATAN and KTANTAN are the same up to the key schedule.
- In KTANTAN, the key is fixed and cannot be changed!

Block Cipher – HW perspective



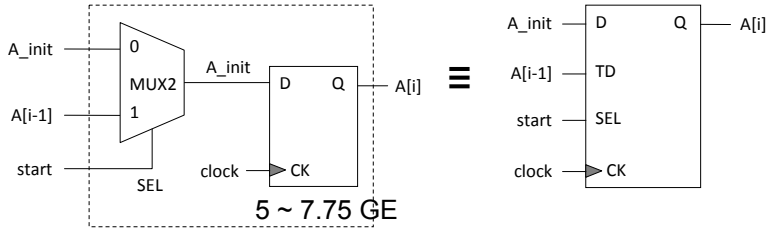
Design Rationale – Memory Issues (1)

- For a more compact cipher, a larger ratio of the area is dedicated for storing the intermediate values and key bits.

Cipher	Block [bits]	Key [bits]	Technology [μm]	Size [GE]	Memory [%]
AES-128 [8]	128	128	0.35	3400	60
AES-128 [10]	128	128	0.13	3100	48
HIGHT [12]	64	128	0.25	3048	49
mCrypton [15]	64	64	0.13	2420	26
DES [19]	64	56	0.18	2309	63
DESL [19]	64	56	0.18	1848	79
PRESENT-80 [4]	64	80	0.18	1570	55
PRESENT-80 [20]	64	80	0.35	1000	≥80

Design Rationale – A Story of a Single Bit

- Assume we have a parallel load of the key and the plaintext.
- A single Flip-Flop has no relevance – MUXes need to be used.
- 2to1 MUX + FF = Scan FF: Beneficial both for area and power.

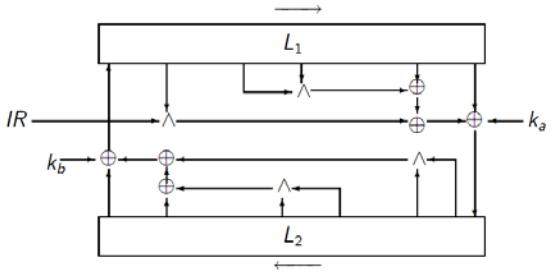


7.25 ~ 13.75 GE 6.25 ~ 11.75 GE

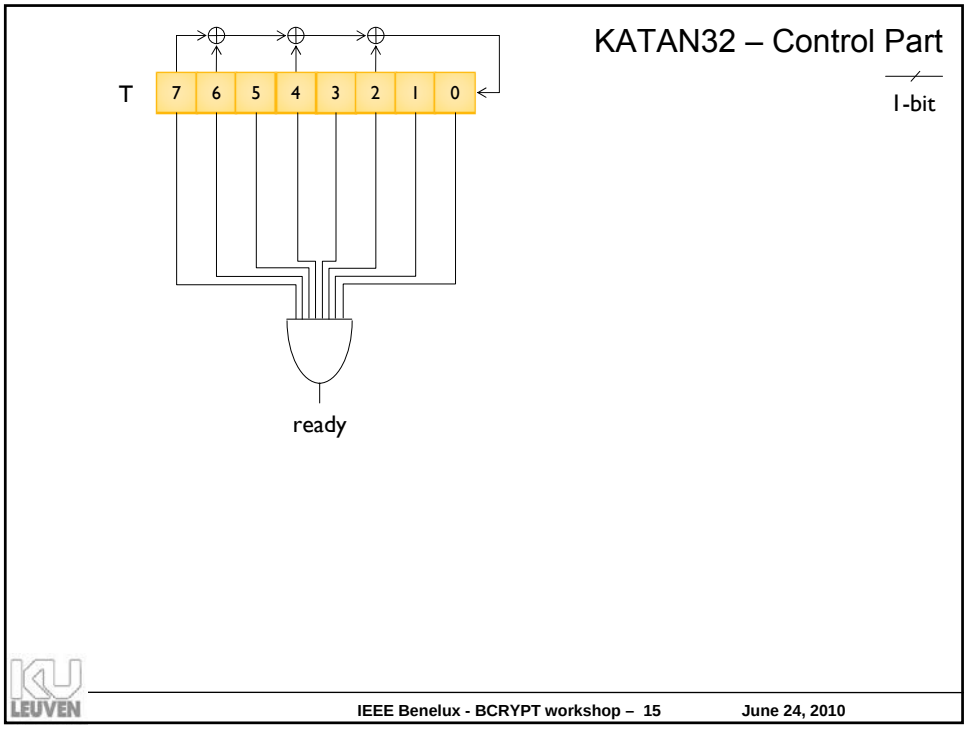
- (64 + 80 + 8) × 6.25 = 950 GE ☺

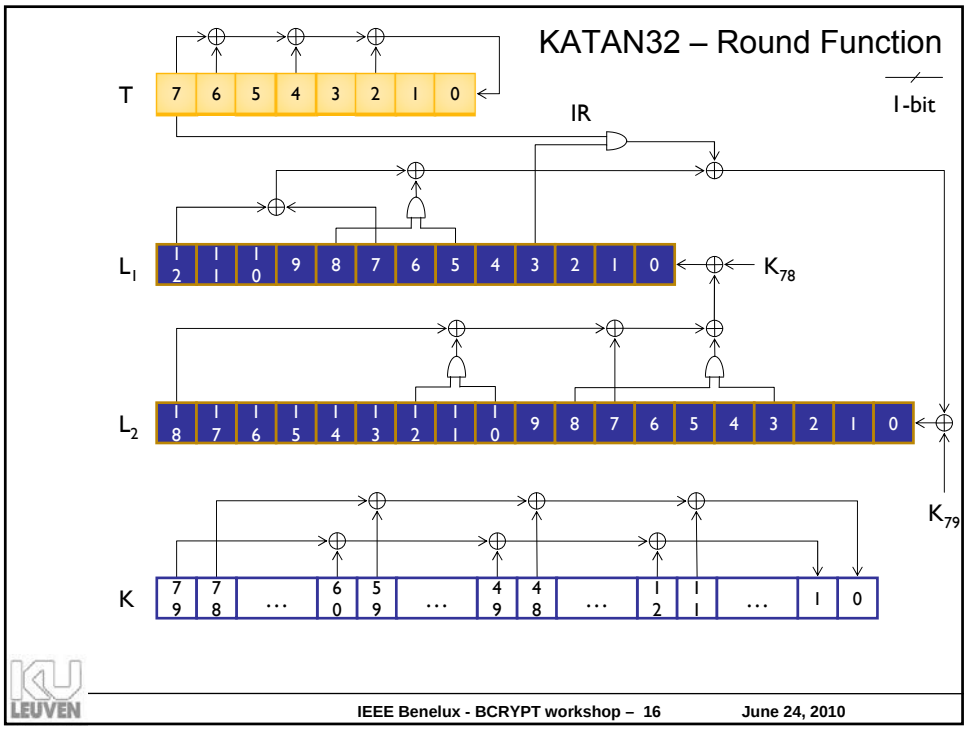
Design Rationale – Control Part

- How to control such a simple construction?



- *IR* stands for *Irregular update Rule*.
- We basically need a counter only. Can it be simpler than that?
- Let the LFSR that is in charge of *IR* play the role of a counter.


KATAN32 – Control Part
1-bit
T
7 6 5 4 3 2 1 0
ready
IEEE Benelux - BCRYPT workshop – 15
June 24, 2010


KATAN32 – Round Function
1-bit
T
7 6 5 4 3 2 1 0
IR
L1
12 11 10 9 8 7 6 5 4 3 2 1 0
K78
L2
18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
K79
K
79 78 … 60 59 … 49 48 … 12 11 … 1 0
IEEE Benelux - BCRYPT workshop – 16
June 24, 2010

Implementation Results

- All designs are synthesized with Synopsys Design Vision version Y-2006.06, using UMC 0.13µm Low-Leakage CMOS library.

Cipher	Block [bits]	Key [bits]	Memory/bit [GE]	Throughput* [Kbps]	Size [GE]
KATAN32	32	80	6.18	12.5	802
KATAN48	48	80	6.19	18.8	927
KATAN64	64	80	6.15	25.1	1054
KTANTAN32	32	80	6.10	12.5	462
KTANTAN48	48	80	6.14	18.8	588
KTANTAN64	64	80	6.17	25.1	688

* Throughput is estimated for frequency of 100 kHz.

SHA3 – competition: how not to do it

"Flexibility" Requirements

The draft minimum acceptability requirements for candidate hash algorithms are:

A.1 The algorithm must be publicly disclosed and available on a worldwide, non-exclusive, royalty-free basis.

A.2 The algorithm must be implementable in a wide range of hardware and software platforms.

A.3 The algorithm must support 224, 256, 384, and 512-bit message digests, and must support a maximum message length of at least 264 bits.

- Wide range of platforms
- Wide range of message digests

[of course, also security requirements]

SHA-3: "cost" requirements

Computational efficiency essentially refers to the throughput of an implementation. NIST will use the

C.2.2 Memory requirements: The memory required for hardware and software implementations of the candidate algorithm will be considered during the evaluation process.

Memory requirements will include such factors as gate counts for hardware implementations, and code size and RAM requirements for software implementations.

- Power consumption?
- Energy to hash one message?

SHA3- results

- NIST asks for a Swiss knife

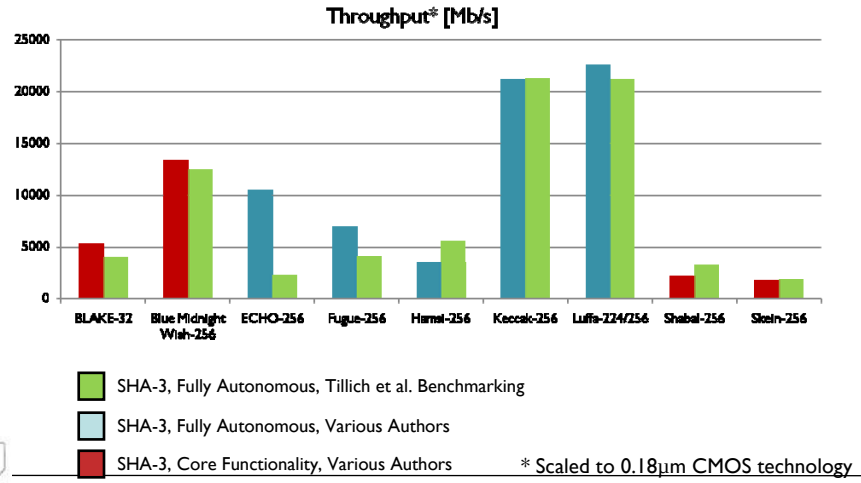
Bread knife

Surgeon's knife

- But often you need a specialized knife
- Certainly for embedded applications

High-Throughput Implementations

http://ehash.iaik.tugraz.at/wiki/SHA-3_Hardware_Implementations



* Scaled to 0.18µm CMOS technology

SHA 3 gate counts

Candidate	State Size [bit]	Total Memory [bit]	Total Memory† [GE]	Total Area [GE]
BLAKE	512	768	4,608	9,890 []
BMW	1,040	1,536	9,216	N/A
CubeHash	512	1,024	6,144	7,630 []
ECHO	2,048	2,560	15,360	82,800 []
Fugue	960	960	5,760	59,220 []
Grøstl	512	1,024	6,144	14,620 []
Hamsi	512	768	4,608	N/A
JH	1,024	1,024	6,144	N/A
Keccak	1,600	1,600	9,600	N/A
Luffa	768	768	4,608	18,260 []
Shabal	1,408	1,408	8,448	23,320 []
SHAvite-3	896	1,024	6,144	N/A
SIMD	512	3,072	18,432	N/A
Skein	256	768	4,608	12,890 []

Estimates for versions with 256-bit digest size are given.
† We estimate the size of a single flip-flop to be 6 GE.

[slide courtesy: Miroslav Knežević]

SHA3 conclusion

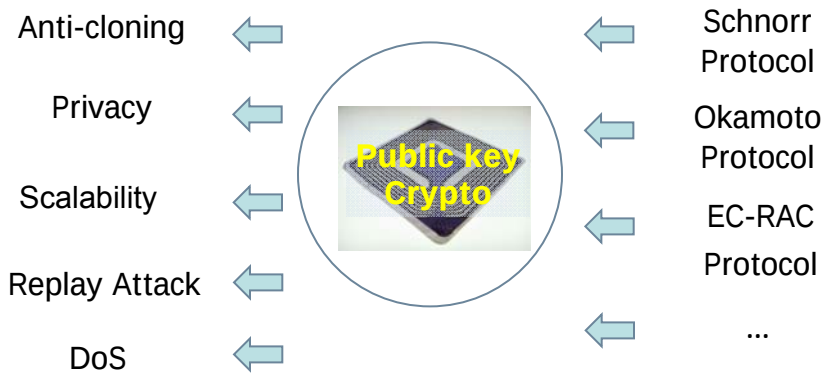
- SHA3 Hash functions are HUGE compared to:
- Secret key algorithms
  - AES: from 3000 gates and up
  - KATAN: around 1000 gates
- Public key algorithms
  - ECC: around 10.000 gates
- Throughput similar to
  - High speed secret key implementations

NEED: domain specific hash functions

Public Key: ECC for RFID

[slide courtesy:
Yong Ki Lee,
Lejla Batina]

Challenge 1: security problems

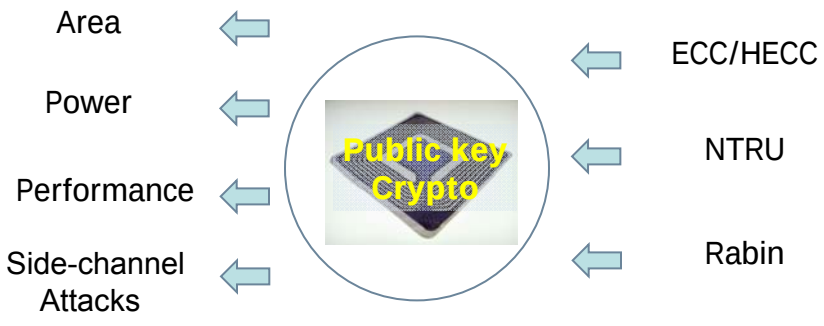


Challenge 1: Security problems

- Current RFID standards:
  - No security
  - Or simple self-destruct password (8 to 32 bits)
- Security challenges RFID:
  - Anti-cloning (make it difficult to 'copy' RFID)
  - Replay attack (query the tag and reuse that info)
  - 'Tracking' attacks => privacy problems
  - Scalability: security for large sets of tags
  - Backward/forward un-traceability

Needs Public key

Challenge 2: design constraints

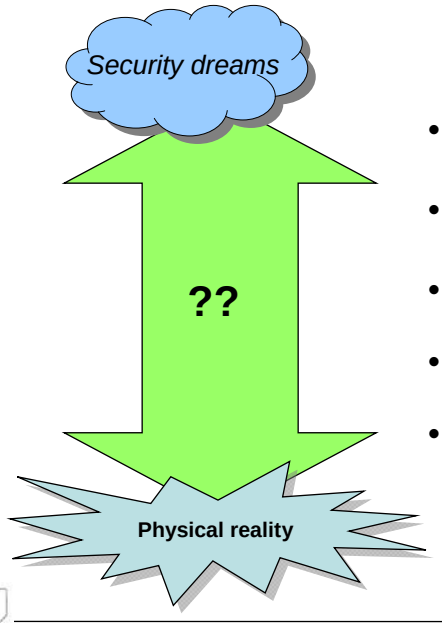


Challenge 2: constraints

Passive RFID tag:

- Area: less than 20.000 gates
- Low Power: total budget varies from 50 to 100 microWatt
- Budget for crypto: less than 15 microWatt!
- Clock frequency: factor of 13.56 MHz
- Execution time target: one point multiplication less than 250 msec.

Design Steps

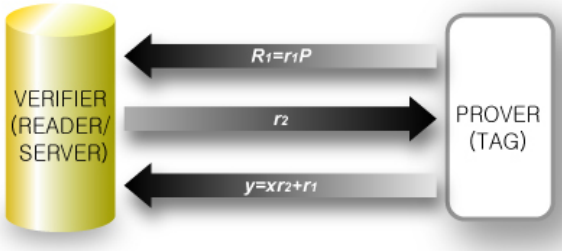


- Step 1: protocols
- Step 2: algorithms
- Step 3: arithmetic
- Step 4: processor
- Step 5: circuits

Step 1: Solutions with Asymmetric-key Algorithms

- Conventional public-key authentication
  - Schnorr protocol, Okamoto Protocol
  - Vulnerable against the tracking attack
- GPS scheme
  - A variant of Schnorr protool
  - Secure transfer of a tag's ID is not solved
- Rabin Encryption
  - Requires a large key size and transmission
  - A compact architecture : WiSec'09(Feldhofer,Oren)

A General EC Authentication Protocol (Schnorr protocol)



$$\{R_1 - yP\} \times r_2^{-1} = \{r_1 P - (xr_2 + r_1)P\} \times r_2^{-1}$$
$$= -xr_2 P \times r_2^{-1} = -xP$$

- A tag's public key can be derived using exchanged messages

  => tracking attack

Observation for RFID Protocols?

- Minimize the computation load on tags
  - We need to transfer computation load to the reader/server as much as possible
- We cannot just transfer ID of a tag
  - A tag's ID is what we need to keep in secret to avoid tracking
- The protocol is a "many to one" protocol
  - A tag's public key (xP) does not need to be publicly known
  - It can be securely stored and used in the server

Step 2: EC based Security Processor

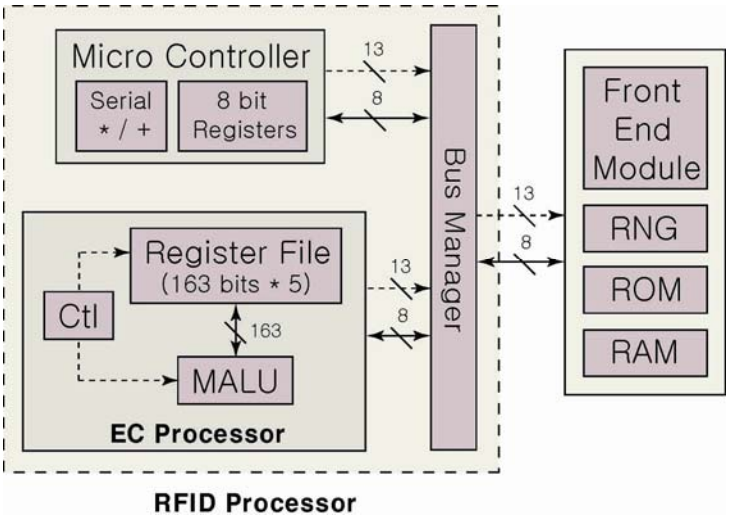
- Operations we need (e.g. EC-RAC)
  - Modular Operation
    - Modular Multiplication: $r_{s1} \bullet x_1 \pmod{n}$
    - Modular Addition: $r_{t1} + r_{s1} x_1 \pmod{n}$

  => *Perform on a 8-bit specialized Micro-Controller*
  - EC Point multiplication
    - $r_{t1} \bullet P$, $(r_{t1} + r_{s1} x_1) \bullet Y$

  => *Perform on a 163 bit Elliptic Curve co-processor*

8 bit versus 163 bit ?? modulo operations are less frequent and not time critical, hence multiplex mod operations

Overall architecture



Step 3: EC Point Multiplication

k•P
Scalar Multiplication
→ Montgomery Algorithm
≈600 GE (Control)

P+Q, 2•P
Point Addition, Point Doubling
→ Lopez-Dahab Algorithm
≈1.2k GE (Control)
+ 6×163 registers

a+b, a•b, a^2
Modular Arithmetic Operation (Addition, Multiplication, Squaring)
→ Sakiyama Modular ALU
≈900 GE (Control)
+ 3×163 registers

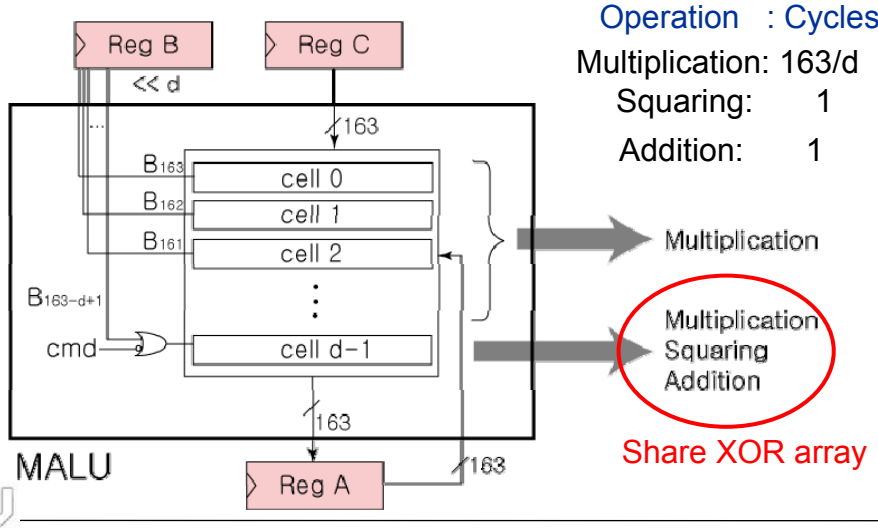
Total Area = 2.7 GE (Control) + 9×163 registers **≈ 80% !!**

* GE: Gate Equivalent (a 2-input NAND)

Step 4: Optimization Approach

- Reduce Registers: 9→5 (4 registers reduction)
  - Common Z-coordinate system : 1 register ↓
  - Redesign Modular ALU : 1 register ↓
  - Register reuse : 2 registers ↓
    - 'Point Add/Dbl algorithm' and 'Modular ALU'
- Reduce Multiplexer Complexity
  - A special Circular Shift Register File
  - Extra 30% reduction in the register file
- Side Channel Resistant

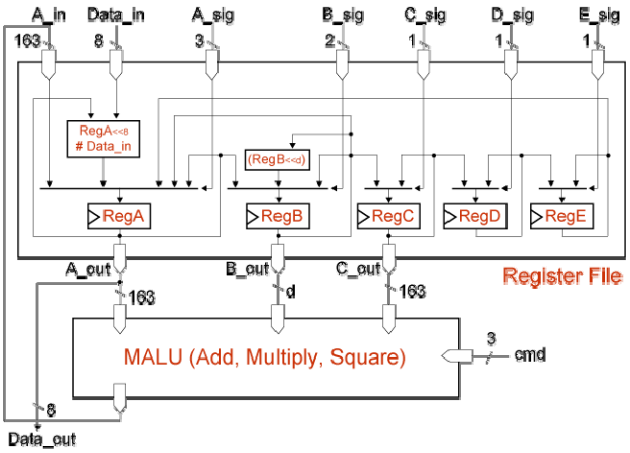
Modular ALU (MALU)



Operation : Cycles
Multiplication: 163/d
Squaring: 1
Addition: 1

Share XOR array

Circular Shift Register



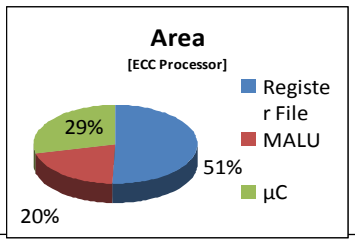
Cost: need more cycles to get data in correct register …
Overall cost: less than 2% compared to point multiplication

Register File Management: shift example

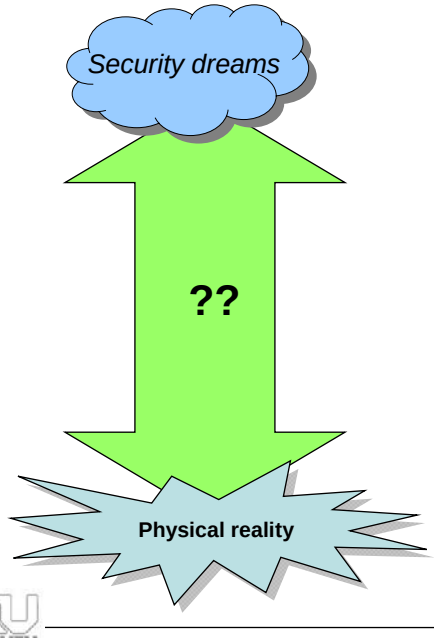
Step	Field Operation	RegA	RegB	RegC	RegD	RegE	cycles	
(1)	Initial	X_2	X_1	X_1	X_2	Z	–	
(2)	1. $T_2 \leftarrow X_1 + X_2$	T_2	X_1	X_1	X_2	Z	1	Add
(3)	2. $T_2 \leftarrow T_2^2$	T_2	X_1	X_1	X_2	Z	1	Square
(4)		X_1	T_2	X_1	X_2	Z	1	Swap
(5)		Z	X_1	T_2	X_2	Z	1	Shift
(6)		X_1	Z	T_2	X_2	X_2	1	Shift
(7)		X_1	X_1	Z	T_2	X_2	1	Shift
(8)	3. $Z \leftarrow Z \cdot X_1$	Z	X_1	–	T_2	X_2	$\lceil 163/d \rceil$	Multiply

Estimated numbers

- Results: ECC co-processor that can compute:
  - ECC point multiplications (163 by 4)
  - Scalar modular operations (8 bit processor with redundancy)
- Schnorr (secure ID transfer, but no tracking protection): **one** PM
- More advanced protocols: up to **four** PM on tag
- Technology: 0.13 micron CMOS low power version
- Size: d=4, 14.500 gates,
- Time: 60.000 cycles for one PM
- Clock at 616 KHz, 97 msec for one PM at 22 microWatt

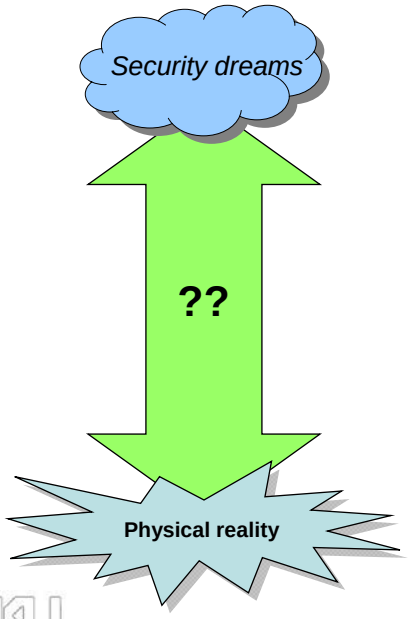


Conclusion



Shows ONE path:

- Protocol design: randomized access
- Public key: ECC many design options
- Architecture: 8 bit micro & 163 EC processor
- Specialized register file
- Full custom layout



Future work:

- Do we have all the required properties covered?
- Can privacy issues be optional, at least for some applications?
- Light weight crypto?
- How to store keys securely?
- Use physical properties: PUF based security ?
- What about side-channel security?